第50卷 第3期
2004年6月

武汉大学学报(理学版)
J. Wuhan Univ. (Nat. Sci. Ed.)

Vol. 50 No. 3
June 2004, 335～338

文章编号:1671-8836(2004)03-0335-04

素域上椭圆曲线密码的高效实现

汪朝晖[1], 陈建华[1], 涂 航[2], 李 莉[2]

(1. 武汉大学 数学与统计学院, 湖北 武汉 430072;
2. 武汉大学 计算机学院, 湖北 武汉 430072)

摘 要: 给出了几个素域上的算术运算和素域上椭圆曲线算术运算的高效实现算法,从而解决了椭圆曲线密码(ECC)实现中影响效率的几个关键算法设计问题,且就ECC加密机制和签名机制的选择提出了建议,最终形成一套高效的素域上ECC加密算法和签名算法的实现方案. 该方案适应多种软件和硬件实现条件,具有较好的通用性.

关 键 词: 椭圆曲线密码; 加密; 签名

中图分类号: TN 918 **文献标识码**: A

1 椭圆曲线密码

椭圆曲线密码(ECC)自1985年由Neal Koblitz[1]和Victor Miller[2]提出以来,由于它比RSA更具优势(更强的安全性、更高的实现效率、更省的实现代价),因此吸引了大批密码学工作者就其安全性和实现方法作了大量的研究,并已逐渐被国际各大标准组织采纳作为公钥密码标准(IEEE P1363、ANSI X9、ISO/IEC、和IETF等).

实现ECC算法,要就基域的选取、基域上算术运算的实现方法、椭圆曲线上算术运算的实现方法和密码机制的确定共4个层次作出综合选择. 这需要综合考虑各种因素,包括安全性要求、实现的性能要求和实现平台的资源条件(如:处理速度、软件或固件环境中的程序空间和数据空间的大小、硬件电路的逻辑门数和功耗等). 过去十几年来,就ECC的实现出现了许多学术文章,但大部分这样的文章或者只论述ECC实现中的某一个层次的算法(如基域上的算术运算或椭圆曲线上的算术运算)或者并没有考虑所有的因素和条件.

本文广泛研究ECC实现中各层次问题,把它们作为一个整体综合考虑,并以软件和固件多种实现平台的资源条件及其上的实验和测试结果为依据,提出一套完整的高效的ECC实现方案,该方案具有良好的通用性和灵活性,适用于多种软/硬件环境下的ECC实现.

2 素域上算术运算的实现

实现ECC之前,首先要确定曲线的基域,目前在国际上主要有两种选择:素数域(记为F_p)或二元扩域(记为F_2^m),本文讨论F_p上ECC的实现方案,该方案对F_2^m上的ECC实现同样有参考价值.

实现F_p上的ECC,需要用到F_p上的加法、减法、乘法(包括平方)、求逆和取模这几种算术运算,根据分析和测试作者发现:模加(加后取模)和模减(减后取模)的运算速度是模乘运算(乘后取模)的10倍以上;求逆运算时间与约70个模乘相当;ECC运算70%左右的时间消耗在模乘运算上. 考虑到上述因素,同时也因为F_p上的模加、模减、乘法、平方和求逆都已有成熟的快速算法,在制定ECC实现方案时,只需考虑使运算尽量少涉及F_p上的求逆,同时要设计高效的对乘积取模的算法.

F_p上的对乘积取模算法的设计与素数p的选取密切相关,如NIST推荐的曲线参数中p的选取可以保证有高效的对乘积取模算法,但对不同的p,其对乘积取模算法各不相同,算法不具备通用性. 在本文给出的方案中,p的选取使得对乘积取模算法

收稿日期: 2003-03-03
基金项目: 国家863计划基金资助项目(2001AA141010)
作者简介: 汪朝晖(1972-),男,博士生,现从事密码学与信息安全的研究. E-mail: Vonsunglon@sina.com

具有良好的通用性,方案中选取的 p 型如:2^m-t(m 为 p 的二进制位长,若处理器的字长为 w,则需保证 $0<t<2^w$),当 m 较大(如 $m=192,244,256$)而 w 较小(如 $w=4,8,16$)时,这样的 p 被称为拟梅森素数,对于这类素数,可以设计出通用的高效的对乘积取模算法.

设 F_p 上两元素的乘积为 X,由于 p 的二进制位长为 m,故可以设 X 的二进制位长为 $2m$(不会超过 $2m$),设 $X=X_H\times 2^m+X_L$(X_H 为 X 的高 m 位,X_L 为 X 的低 m 位),则

$$X \bmod p \equiv (X_H\times 2^m+X_L)\bmod p \equiv (X_H\times 2^m \bmod p+X_L)\bmod p \equiv (X_H\times t+X_L)\bmod p$$

上式表明对二进制位长为 $2m$ 的 X 取模等于对二进制位长为 $m+w+1$ 的 $X_H\times t+X_L$ 取模,又设 $Y=X_H\times t+X_L=Y_H\times 2^m+Y_L$($Y_H$ 为 Y 的高 $w+1$ 位,Y_L 为 Y 的低 m 位),则

$$Y \bmod p \equiv (Y_H\times 2^m+Y_L)\bmod p \equiv (Y_H\times 2^m \bmod p+Y_L)\bmod p \equiv (Y_H\times t+Y_L)\bmod p$$

所以对二进制位长为 $m+w+1$ 的 Y 取模等于对二进制位长为 $m+1$ 的 $Y_H\times t+Y_L$ 取模,故有算法 1.

算法 1 对乘积取模

输入:$X(=X_H\times 2^m+X_L)$,$t(p=2^m-t)$

输出:$X \bmod p$

定义二进制位长为 $m+w+1$ 的数 Y;

$Y\leftarrow X_H\times t+X_L$;

将 Y 分为两部分 Y_H 和 Y_L 使得 $Y=Y_H\times 2^m+Y_L$;

$Y\leftarrow Y_H\times t+Y_L$;

当 $Y>p$ 时循环做 $Y\leftarrow Y-p$;

输出 Y.

可见算法 1 对型如 2^m-t 的 p 是通用的,不论 p 的二进制位长 m 为多少.经测试,该算法的运行速度约为乘法运算的 1/10,与针对 NIST 推荐的 p 设计的特殊取模算法相当,而远远快于其他的取模算法.表 1 列出了作者实现的该算法与其他相关算法的运算速度对比(测试环境:Intel PIII 800 MHz CPU,Windows98,ANSI 标准 C 语言编程,p 长 192 位).

表 1 算法 1 及其他相关算法的性能对照表

算法	算法 1	Barrett 取模	NIST 特殊 p 取模	标准乘法
速度(μs/次)	0.231	3.674	0.238	2.236

3 椭圆曲线上算术运算的实现

一般 F_p 上的椭圆曲线具有 Weierstrass 方程:$y^2=x^3+ax+b$,本文中的 ECC 实现方案选择型如 $y^2=x^3-3x+b$ 的椭圆曲线(即取 $a=-3$),这样使得椭圆曲线上点的运算可以更有效些.

若曲线上有两仿射坐标点 $P_1=(x_1, y_1)$ 和 $P_2=(x_2, y_2)$,则其和 $P_3=P_1+P_2=(x_3, y_3)$ 可由下述公式组(1)求得:

$$\begin{cases} x_3 = r^2-x_1-x_2 \\ y_3 = r(x_1-x_3)-y_1 \end{cases} \tag{1}$$

式中:

$$r=\begin{cases}(3x_1-3)/(2y_1), & \text{当 } P_1=P_2 \text{ 时}\\ (y_1-y_2)/(x_1-x_2), & \text{当 } P_1\neq P_2 \text{ 时}\end{cases}$$

可以看出,若用公式组(1)计算椭圆曲线上的点加,则每次点加都需一次 F_p 上的求逆运算,这是相当耗时的.为避免在点加中使用求逆,作者使用雅可比投影坐标(x, y, z)表示椭圆曲线上的点,其对应的仿射坐标点为$(x/z^2, y/z^3)$,则对于椭圆曲线上点的加法,可以得到以下无需求逆的几组运算公式.

$2(x_1, y_1, z_1)=(x_1, y_1, z_1)+(x_1, y_1, z_1)=(x_3, y_3, z_3)$(投影坐标点的加倍):

$$\begin{cases} A = 4x_1y_1^2 \\ B = 8y_1^4 \\ C = 3(x_1-z_1^2)(x_1+z_1^2) \\ x_3 = C^2-2A \\ y_3 = C(A-x_3)-B \\ z_3 = 2y_1z_1 \end{cases} \tag{2}$$

$(x_1, y_1, z_1)+(x_2, y_2)=(x_3, y_3, z_3)$(投影坐标点加仿射坐标点):

$$\begin{cases} A = x_2z_1^2 \\ B = y_2z_1^3 \\ C = A-x_1, D = B-y_1 \\ x_3 = D^2-(C^3+2x_1C^2) \\ y_3 = D(x_1C^2-x_3)-y_1C^3 \\ z_3 = z_1C \end{cases} \tag{3}$$

$(x_1, y_1, z_1)+(x_2, y_2, z_2)=(x_3, y_3, z_3)$(投影坐标点加投影坐标点):

$$\begin{cases} A = x_1z_2^2 \\ B = y_1z_2^3 \\ C = x_2z_1^2-A \\ D = y_2z_1^3-B \\ x_3 = D^2-2AC^2-C^3 \\ y_3 = D(AC^2-x_3)-BC^3 \\ z_3 = z_1z_2C \end{cases} \tag{4}$$

运用上述公式组,可以设计出椭圆曲线上点乘算法,即计算 kP 的算法(其中 k 为一大整数,P 为椭圆曲线上的一点).当 P 为不定点时,计算 kP 是复杂且耗时的;当 P 为固定点时,可以先对 P 进行预计算,利用预计算得到的数据加速 kP 的计算过程.

通过对各种点乘算法的详细分析和在各种软/硬件环境下的大量测试,作者在方案中选择下面的算法来计算不定点的点乘,该算法与计算模幂的滑动窗口法[3]类似:

算法 2　不定点的点乘

输入:$P=(x, y)$,k

输出:$Q=kP$

1) 计算 k 的 NAF(non_adjacent_form)表示形式$(k_m, k_{m-1}, \cdots, k_1, k_0)$,$k_i \in \{1, 0, -1\}$;

2) 用公式组(2)、(3)和(4)计算用投影坐标表示的点 $6P$、$7P$、$8P$、$9P$ 和 $10P$;

3) $Q \leftarrow O$(称 O 为零点,用投影坐标(0, 1, 0)表示);

4) i 从 m 开始,当 $i \geqslant 3$ 时循环做.若 k_i 不为 0,则{用公式组(2)计算 $Q \leftarrow 16Q$;$n \leftarrow 2^3 k_i + 2^2 k_{i-1} + 2k_{i-2} + k_{i-3}$;若 $n>0$ 则用公式组(4)计算 $Q \leftarrow Q + |n|P$,否则用公式组(4)计算 $Q \leftarrow Q - |n|P$;$i \leftarrow i-3$},否则用公式组(2)计算 $Q \leftarrow 2Q$;$i \leftarrow i-1$;

5) 当 $i \geqslant 0$ 时循环做.用公式组(2)计算 $Q \leftarrow 2Q$;若 $k_i>0$ 则用公式组(3)计算 $Q \leftarrow Q+P$;若 $k_i<0$ 则用公式组(3)计算 $Q \leftarrow Q-P$;$i \leftarrow i-1$;

6) 如果需要,将 Q 转换为仿射坐标点;输出 Q.

文献[4]提出了一种加速模幂运算的预计算方法,作者将之稍作修改后,用于以预计算来加速点乘运算.设大整数的二进制表示串排列为 w 行 d 列,定义$\langle a_{w-1}, \cdots, a_1, a_0\rangle P = a_{w-1} 2^{(w-1)d} P + \cdots + a_2 2^{2d} P + a_1 2^d P + a_0 P (a_i \in \{0,1\})$,设 $e=[(d+1)/2]$,则对于椭圆曲线上的一固定点 P 和所有的 $\langle a_{w-1}, \cdots, a_1, a_0\rangle \in \{0,1\}^w$,可以预计算 $\langle a_{w-1}, \cdots, a_1, a_0\rangle P$ 和 $2^e\langle a_{w-1}, \cdots, a_1, a_0\rangle P$,预计算得到的点用仿射坐标表示.利用预计算得到的点,作者设计出下面的算法来加速点乘运算.

算法 3　固定点的点乘

输入:预计算得到的所有点 $\langle a_{w-1}, \cdots, a_1, a_0\rangle P$ 和 $2^e\langle a_{w-1}, \cdots, a_1, a_0\rangle P$,$k$

输出:$Q=kP$

1) $Q \leftarrow O$(称 O 为零点,用投影坐标(0, 1, 0)表示);

2) 将 k 的二进制表示串按从低到高的顺序排列成 w 行,即设 $k=k^{w-1} \| \cdots \| k^1 \| k^0$($k^i$ 均为长为 d 的二进制串),设 k_i^j 表示 k^j 的第 i 位;

3) i 从 $e-1$ 开始,当 $i \geqslant 0$ 时循环做.用公式组(2)计算 $Q \leftarrow 2Q$;用公式组(3)计算 $Q \leftarrow Q + \langle k_i^{w-1}, \cdots, k_i^1, k_i^0\rangle P$;用公式组(3)计算 $Q \leftarrow Q + 2^e\langle k_{i+e}^{w-1}, \cdots, k_{i+e}^1, k_{i+e}^0\rangle P$;$i \leftarrow i-1$;

4) 如果需要,将 Q 转换为仿射坐标点;输出 Q.

显然 w 越大,预计算得到的点就越多($2^{(w-1)}-2$ 个),需要的存储空间就越多,算法 3 的运行速度就越快.读者可以根据实际环境的存储器资源情况调整 w 的大小,以保障算法 3 的通用性.

表 2 列出了作者实现的算法 2、算法 3 与其他相关算法的运算速度对比(测试环境:TI 200 MHz DPS TMS320VC5510,汇编语言编程,p 长 192 位).

表 2　算法 2,算法 3 及其他相关算法的性能对照表

算法	算法 2	采用标准投影坐标的算法 2	算法 3	采用分段预计算的固定点乘
速度(ms/次)	4.366	5.183	0.838	0.892

至此,本文已经确定了椭圆曲线上算术运算的高效实现方案,在此上选择合适的密码算法,就形成一套完整的 ECC 高效实现方案.

4　密码机制的选择

在 ECC 出现之初,人们一般将 ElGamal 加密机制[5]平行移植到椭圆曲线上,做为其上的加密机制,因而就出现了明文的嵌入问题,有很多密码学工作者就此问题做了研究,始终找不到一个针对一般椭圆曲线的确定性的明文嵌入算法.人们大量采用的是概率性的明文嵌入算法,经作者测试这类明文嵌入算法的运行时间约占 ECC 加密算法运行总时间 40%,且代码量要增加约 40%,也就是说明文嵌入算法消耗了很大部分的处理器和内存资源,给 ECC 的软/硬件实现带来了麻烦.

彻底解决明文嵌入问题源自椭圆曲线上的变型 ElGamal 加密机制的出现,如 ECMV 机制和 ECIES 机制,这类加密机制将秘密信息对明文的扰动运算放在椭圆曲线上的点运算以外,从而使明文不需要嵌入到椭圆曲线上的点上去.这类机制的出现使得

ECC的实现效率可以极大提高,且实现代价大幅下降,从而加速了ECC的推广和应用.在作者设计的ECC实现方案中,加密机制采用不需要明文嵌入的椭圆曲线上的变型ElGamal机制,既使ECC加密机制的安全性得到有效保障,又针对椭圆曲线上算术运算的特征,获得很高的性能/代价比.

人们很容易将ElGamal型签名机制[5]移植到椭圆曲线上,从而获得ECC签名机制如ECDSA[6,7]和ECNR[8]等.由于ElGamal型签名机制有很多,选择范围较广,在作者设计的ECC实现方案中,签名机制采用不需要求逆的ElGamal型签名机制,经过作者大量测试表明,这类签名算法的实现效率要较其他ElGamal型签名机制约高出10%.

5 结束语

本文分析了影响ECC实现效率的关键因素,针对影响效率的几个关键性的算术运算给出了高效实现算法,并就椭圆曲线上的加密机制和签名机制的选择提出建议,从而形成了一套完整的F_p上ECC实现的高效方案,该方案经作者在大量软/硬件环境下的实践证明,是高效且具有良好的通用性的.作者进一步的研究工作已放在该实现方案的安全性上,并已有成果保证该方案的安全性.

参考文献:

[1] Koblitz N. Elliptic Curve Cryptosystems[J]. *Mathematics of Computation*, 1987, **48**: 203-209.

[2] Miller V. Uses of Elliptic Curves in Cryptography [A]. *Advances in Cryptology-Crypto'85*, *LNCS*218 [C]. New York: Springer-Verlag, 1986, 417-426.

[3] Gordon D. A Survey of Fast Exponentiation Methods [J]. *Journal of Algorithms*, 1998, **27**: 129-146.

[4] Lim C, Lee P. More Flexible Exponentiation with Precomputation[A]. *Advances in Cryptology-Crypto'94*, *LNCS*839[C]. New York: Springer-Verlag, 1994, 95-107.

[5] ElGamal T. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms[J]. *IEEE Transactions on Information Theory*, 1985, **31**: 469-472.

[6] National Institute of Standards and Technology. Digital Signature Standard[S]. FIPS Publication 186, 1993.

[7] Johnson D, Menezes A. The Elliptic Curve Digital Signature Algorithm (ECDSA)[R]. Waterloo: Dept. of C&O, University of Waterloo, 1999.

[8] Nyberg K, Rueppel R A. A New Signature Scheme Based on the DSA Giving Message Recovery[A]. *1st ACM Conf. on Computer and Communication Security*[C]. New York: ACM Press, 1993.

Efficient Implementation of Elliptic Curve Cryptosystem over Prime Fields

WANG Zhao-hui[1], CHEN Jian-hua[1], TU Hang[2], LI Li[2]

(1. School of Mathematics and Statistics, Wuhan University, Wuhan 430072, Hubei, China;
2. School of Computer, Wuhan University, Wuhan 430072, Hubei, China)

Abstract: The authors present efficient algorithms for implementations of arithmetic operations in prime fields and elliptic curves over such fields, thus solved the issues of designing the key algorithms which lay great emphasis on the efficiencies of elliptic curve cryptosystems. The authors also gave suggestions about how to selecting efficient encryption schemes and signature schemes for elliptic curve cryptosystems. All those outcomes form a scenario of practical implementations of elliptic curve encryption algorithms and elliptic curve signature algorithms over prime fields, which is adaptive to many a software or hardware implementation environment and has pretty good generality.

Key words: elliptic curve cryptosystems; encryption; signature